NAME

BIO_f_base64 − base64 BIO filter

SYNOPSIS

```
#include <openssl/bio.h>
#include <openssl/evp.h>

const BIO_METHOD *BIO_f_base64(void);
```

DESCRIPTION

BIO_f_base64() returns the base64 BIO method. This is a filter BIO that base64 encodes any data written through it and decodes any data read through it.

Base64 BIOs do not support **BIO_gets()** or **BIO_puts()**.

For writing, by default output is divided to lines of length 64 characters and there is a newline at the end of output. This behavior can be changed with **BIO_FLAGS_BASE64_NO_NL** flag.

For reading, the first line of base64 content should be at most 1024 bytes long including newline unless the flag **BIO_FLAGS_BASE64_NO_NL** is set. Subsequent input lines can be of any length (i.e., newlines may appear anywhere in the input) and a newline at the end of input is not needed.

Also when reading, unless the flag **BIO_FLAGS_BASE64_NO_NL** is set, initial lines that contain non−base64 content (whitespace is tolerated and ignored) are skipped, as are lines longer than 1024 bytes. Decoding starts with the first line that is shorter than 1024 bytes (including the newline) and consists of only (at least one) valid base64 characters plus optional whitespace. Decoding stops when base64 padding is encountered, a soft end−of−input character (−, see **EVP_DecodeUpdate** (3)) occurs as the first byte after a complete group of 4 valid base64 characters is decoded, or when an error occurs (e.g. due to input characters other than valid base64 or whitespace).

If decoding stops as a result of an error, the first **BIO_read** (3) that returns no decoded data will typically return a negative result, rather than 0 (which indicates normal end of input). However, a negative return value can also occur if the underlying BIO supports retries, see **BIO_should_read** (3) and **BIO_set_mem_eof_return** (3).

BIO_flush() on a base64 BIO that is being written through is used to signal that no more data is to be encoded: this is used to flush the final block through the BIO.

The flag **BIO_FLAGS_BASE64_NO_NL** can be set with **BIO_set_flags()**. For writing, it causes all data to be written on one line without newline at the end. For reading, it removes all expectations on newlines in the input data.

NOTES

Because of the format of base64 encoding the end of the encoded block cannot always be reliably determined.

RETURN VALUES

BIO_f_base64() returns the base64 BIO method.

EXAMPLES

Base64 encode the string "Hello World\n" and write the result to standard output:

```
BIO *bio, *b64;
char message[] = "Hello World \n";

b64 = BIO_new(BIO_f_base64());
bio = BIO_new_fp(stdout, BIO_NOCLOSE);
BIO_push(b64, bio);
BIO_write(b64, message, strlen(message));
BIO_flush(b64);

BIO_free_all(b64);
```

Read base64 encoded data from standard input and write the decoded data to standard output:

```
BIO *bio, *b64, *bio_out;
char inbuf[512];
int inlen;

b64 = BIO_new(BIO_f_base64());
bio = BIO_new_fp(stdin, BIO_NOCLOSE);
bio_out = BIO_new_fp(stdout, BIO_NOCLOSE);
BIO_push(b64, bio);
while ((inlen = BIO_read(b64, inbuf, 512)) > 0)
    BIO_write(bio_out, inbuf, inlen);

BIO_flush(bio_out);
BIO_free_all(b64);
```

BUGS

The hyphen character (−) is treated as an ad hoc soft end−of−input character when it occurs at the start of a base64 group of 4 encoded characters.

This heuristic works to detect the ends of base64 blocks in PEM or multi−part MIME, provided there are no stray hyphens in the middle input. But it is just a heuristic, and sufficiently unusual input could produce unexpected results.

There should perhaps be some way of specifying a test that the BIO can perform to reliably determine EOF (for example a MIME boundary).

It may be possible for **BIO_read** (3) to return zero, rather than −1, even if an error has been detected, more tests are needed to cover all the potential error paths.

SEE ALSO

BIO_read (3), **BIO_should_read** (3), **BIO_set_mem_eof_return** (3), **EVP_DecodeUpdate** (3).

COPYRIGHT